NODE.JS NIGHTS

STRV

TESTING

David Ruzicka, Backend Developer at STRV

STRV

Why to test?

“Without some sort of testing we can never be sure if our code is working or not.“

HOW TO TEST?

Manually

- Usable only on simplest of projects
- Difficult to maintain
- Doesn’t scale
- Doesn’t support CI/CD

Automatically

- Usable for every project
- Less difficult to maintain
- Does scale
- Supports CI/CD
- Can provide much more added value
- Test frameworks (Mocha, Jest, ...)

TYPES OF AUTOMATIC TESTING

- **Unit testing**
- **Integration testing**
- End-to-end testing
- And there’s a lot more...

UNIT TESTING

UNIT TESTING

- Making sure that individual parts of the code works correctly
- Unit - as a smallest unit of code (function)
- Units are tested in isolation from other parts
- Enforces developers to write decoupled components

INTEGRATION TESTING

INTEGRATION TESTING

- Making sure that individual parts put (integrated) together works correctly
- Designed to expose faults of exposed interfaces
- In terms of API, we are testing whole endpoints

COVERAGE

COVERAGE

- Gives you some measure of how good your tests are
- Discovers your blind spots in your tests
- Can be integrated into your CI flow
- TIP: Always use option to generate coverage for all files!
- TIP: Don't strive for 100% coverage

MOCKING

MOCKING

- Reducing complexity by replacing some parts of the code with it’s imitation
- Sinon
- mocking/stubbing/faking…
- Nock

TDD

TDD - Test-driven development

- Write tests first, implementation later
- Useful when you have clearly defined task
- Not so useful in other cases (big projects)

HOMEWORK

STRV

QUESTIONS

STRV